tot Metropolia

Erkka Peltoniemi

Windows Phone 7 -sovelluskehitys

Metropolia Ammattikorkeakoulu
insinööri (AMK)
Tietotekniikka
Opinnäytetyö
15.3.2011

Metropolia

Tiivistelmä

Tekijä(t) Otsikko Sivumäärä Aika	Erkka Peltoniemi Windows Phone 7 -sovelluskehitys 39 sivua + 1 liite 15.3.2011
Tutkinto	insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Juha Pekka Kämäri Lehtori Jorma Räty

Microsoft toi 2010-loppupuolella markkinoille uuden käyttöjärjestelmänsä, joka uudisti Microsoftin mobiilitarjonnan. Windows Phone 7 -käyttöjärjestelmä on kehitetty kokonaan alusta asti, joten sillä ei ole enää huonomaineisen Mobile 6.5 -käyttöjärjestelmän kanssa riippuvuutta tai painolastia.

Tämän opinnäytetyön tavoitteena on tutkia Phone 7 -sovelluskehitystyökaluja jotka julkaistiin kehittäjille maaliskuussa 2010. Sovelluskehitystyökalut jakaantuvat kahteen erilaiseen osakokonaisuuteen, joista XNA on suunnattu yksinomaan mobiilipelikehitykseen ja Silverlight puolestaan sovellusalustaksi.

Opinnäytetyössä tutkitaan Phone 7 -sovelluskehityksen lisäksi myös Microsoftin kehittämää Marketplace -mobiilikauppapaikkaa, jonka avulla sovelluskehittäjät voivat levittää sovelluksiaan ympäri maailmaa. Prosessiin, jota seuraamalla kehittäjä voi julkaista sovelluksensa kauppapaikalle, liittyy Microsoftin suorittamia erilaisia laatua ja tietoturvaa parantamaan pyrkiviä vaiheita.

Avainsanat	Windows Phone 7, mobiili, sovelluskehitys, Microsoft, Silverlight, XNA

Metropolia

Abstract

Author(s) Title Number of Pages Date	Erkka Peltoniemi Windows Phone 7 -sovelluskehitys 39 pages + 1 appendices 15 March 2011
Degree	Bachelor of Science in Computer Engineering
Degree Programme	Computer Science
Specialisation option	Software Engineering
Instructor(s)	Juha Pekka Kämäri (Principal Lecturer) Jorma Räty (Lecturer)

At the end of the year 2010 Microsoft released a new mobile operation system that renewed the company's contribution to the mobile market. Windows Phone 7 is built entirely from scratch and is thereby not dependent on the old Windows Mobile operation system.

The target of this thesis was to give an overview of the Windows Phone 7 software development tools that were released to the community in March 2010. The developers can choose from two different platforms where XNA is targeted for the mobile game industry and Silverlight as a platform for rich media and business applications. The thesis approaches the subject by giving a walkthrough over a published commercial Windows Phone 7 application.

This thesis also provides an overview of Market Place, a Microsoft provided service for publishing and distributing applications for Windows Phone.

Keywords	Microsoft, Windows Phone 7, mobile, software development, Silverlight, XNA

Sisällys

1 Johdanto

Microsoft toi 2010-loppupuolella markkinoille uuden käyttöjärjestelmänsä, joka uudisti Microsoftin mobiilitarjonnan. Windows Phone 7 -käyttöjärjestelmän kehitys on aloitettu kokonaan puhtaalta pöydältä, joten sillä ei ole enää huonomaineisen Mobile 6.5 -käyttöjärjestelmän kanssa riippuvuutta tai painolastia. Täten vanhan Mobile-käyttöjärjestelmän sovellukset eivät myöskään ole yhteensopivia Phone 7 -käyttöjärjestelmän kanssa ja Phone 7:lle kehitettyjä sovelluksiakaan ei voi sellaisenaan ajaa Mobile-käyttöjärjestelmässä.

Tämä raportti tutustuttaa lukijan Phone 7 -käyttöjärjestelmään ja sovelluskehitykseen. Raportissa käydään ensin läpi Microsoftin historia mobiilimarkkinoilla, jonka kautta muodostuu kuva nykytilanteesta ja siitä, miksi Phone 7 -käyttöjärjestelmä on olemassa.

Tämän jälkeen tutustutaan itse Phone 7 -käyttöjärjestelmään, sen peruskonsepteihin ja päätelaitteiden laitevaatimuksiin. Tätä kautta päästään lopulta tutkimaan ja testaamaan Phone 7 -sovelluskehitystyökaluja, jotka julkaistiin sovelluskehittäjille 2010-vuoden alkupuolella. Raportissa tullaan dokumentoimaan oikea Phone 7 -käyttöjärjestelmän päälle kehitetty sovellus, joka julkaistaan Phone Marketplaceen. Marketplacen avulla sovelluskehittäjät voivat levittää sovelluksiaan ympäri maailmaa, ja se on tiiviisti integroitu osaksi Phone 7 -käyttöjärjestelmää.

Raportissa tutustutaan myös prosessiin, jota seuraamalla kehittäjä voi julkaista sovelluksensa kauppapaikalle. Prosessiin liittyy Microsoftin suorittamia erilaisia laatua ja tietoturvaa parantamaan pyrkiviä vaiheita, joilla pyritään pitämään Marketplacen sovellustarjonta laadukkaana ja turvallisena.

2 Microsoftin historia mobiilimarkkinoilla

Microsoftin mobiilimarkkinoiden valloituksen voidaan katsoa alkaneen vuonna 1996, kun yhtiö julkaisi ensimmäisen version Windows CE -käyttöjärjestelmästään. Käyttöjärjestelmä suunniteltiin käytettäväksi pienikokoisissa ja suorituskyvyltään vaatimattomissa kämmentietokoneissa sekä sulautetuissa järjestelmissä. Muistia käyttöjärjestelmä vaati ainoastaan megatavun verran.

Windows CE:n käyttöliittymä suunniteltiin jäljittelemään Windows 95 graafista käyttöliittymää mutta sen käyttöjärjestelmäydin ei kuitenkaan perustunut Windows 95 ytimeen.

Windows CE loi pohjan yhtiön myöhemmin kehittämille Pocket PC- ja Windows Mobile -käyttöjärjestelmille ja siitä on muodostunut yleinen käyttöjärjestelmä erityisesti sulautetuissa reaaliaikaisjärjestelmissä. [1.]

Windows Mobile sai alkunsa vuonna 2000, kun Microsoft julkaisi Pocket PC 2000 -käyttöjärjestelmän. Pocket PC 2000 perustui Windows CE 3.0, ja se oli taaksepäin yhteensopiva yhtiön Palm-Size PC -laitteiden kanssa. Pocket PC:n käyttöliittymä suunniteltiin jäljittelemään mahdollisimman paljon Microsoftin Windows 98 -käyttöliittymää ja sitä ohjattiin enimmäkseen stylus-kynän avustuksella.

Pocket PC 2000 -laitteissa ei vielä ollut varsinaisia matkapuhelintoiminnallisuuksia. Nämä ominaisuudet tulivat mukaan vasta osassa Pocket PC 2002 -laitteista muutamia vuosia myöhemmin. Pocket PC 2002 oli ensimmäinen Microsoftin mobiilikäyttöjärjestelmä, jota hyödynnettiin älypuhelimissa ja sen käyttöliittymä oli samankaltainen Windows XP:n kanssa.

Pocket PC -käyttöjärjestelmien jälkeen julkaistiin vuonna 2003 ensimmäinen Windows Mobile -käyttöjärjestelmä. Windows Mobile 2003 kehitettiin neljä erilaista versiota Pocket PC- ja älypuhelimille. Käyttöjärjestelmän ydin pohjautui Windows CE 4.2 kerneliin. Vuonna 2004 Mobile 2003 julkaistiin vielä päivitetty Windows Mobile 2003 SE -

versio, joka sisälsi tuen uusille näyttöresoluutioille sekä langattoman wlan-vastaanottimen.

Vuonna 2005 saapui jälleen uusi Windows Mobile -sarjan käyttöjärjestelmä. Windows Mobile 5 sisälsi .NET Compact Framework 1.0 SP3 -sovelluskehikon, jolla käyttöjärjestelmässä pystyttiin ajamaan rajoitetusti .NET-ohjelmia. Käyttöjärjestelmä sisälsi myös Persistent Storage -ominaisuuden, jossa Flash-muistin ja perinteisen RAM-muistin saumattomalla yhteiskäytöllä pystyttiin parantamaan käyttöjärjestelmää hyödyntävien laitteiden akun kestoa. [2.]

Mobile 5 ilmestymisen aikoihin Symbian-mobiilikäyttöjärjestelmä oli kaikkein yleisin älypuhelinkäyttöjärjestelmä, ja yli puolet myytävistä älypuhelimista pyöri Symbian OS:lla. Windows Mobile löytyi silloin n. 17 % mobiililaitteista ja oli edellisvuoteen verrattuna kasvattanut markkinaosuuksiaan. Markkinoita jo kauan hallinnut Symbian oli jo kuitenkin aloittanut laskunsa markkinaosuuksissa uusien käyttöjärjestelmien ottaessa tilaa mobiilimarkkinoilta.

Microsoftin seuraava mobiilikäyttöjärjestelmä, Windows Mobile 6, ei käyttöliittymältään poikennut kovin paljon edeltäjästään Mobile 5:sta. Mobile 6 suunniteltiin kuitenkin jäljittelemään ulkoasultaan ja käyttökokemukseltaan Windows Vistaa. Käyttöjärjestelmä julkaistiin vuonna 2007, ja se oli tiukasti integroitu Microsoftin Live ja Exchange -järjestelmiin.

Mobile 6 -käyttöjärjestelmästä tehtiin vielä ennen Windows Phone 7:n tuloa 4 versiopäivitystä, joista viimeinen, Mobile 6.5 tuli ylimääräinen väliversio pelastamaan Phone 7:n myöhästymisen aiheuttamaa kuluttajien pakoa kilpailevien käyttöjärjestelmien, lähinnä Androidin ja iOS:n, leireihin.

Ennen Phone 7 julkaisua 2010-vuoden lopussa Microsoft oli painunut lähes olemattomaksi tekijäksi mobiilimarkkinoilla, koska yhtiön mobiilikäyttöjärjestelmät olivat auttamattoman kömpelöitä ja taipumattomia kilpailijoiden moderneihin käyttöjärjestelmiin verrattuna.

3 Windows Phone 7 -käyttöjärjestelmä

Koska Windows Mobile 6.5 ei enää pystynyt vastaamaan kuluttajien kasvaviin vaatimuksiin mobiilimarkkinoilla, Microsoftilla oli kiire kehittää uusi käyttöjärjestelmä, joka palauttaisi yhtiön uskottavuuden mobiilikäyttöjärjestelmävalmistajana. Windows Phone 7 -käyttöjärjestelmän kehitys olikin varsin nopeaa ja tästä syystä yhtiö päätti mm. katkaista käyttöjärjestelmän taaksepäin-yhteensopivuuden aiempiin Mobile-käyttöjärjestelmiin.

Windows Phone 7 -käyttöjärjestelmä perustuu Windows CE 6.0 -ytimeen, joka mahdollistaa mm. Silverlight-sovellusten ajon ytimen päällä. Silverlight-alusta onkin toinen Phone 7 -käyttöjärjestelmän sovellusalustoista jonka päälle kehittäjät voivat ohjelmoida monipuolisia käyttöliittymäpainotteisia mobiilisovelluksia. Toinen käyttöjärjestelmän tukema sovellusalusta on Microsoft XNA, jolla ohjelmistokehittäjät voivat luoda pelejä jotka hyödyntävät Phone 7 -laitteiden ominaisuuksia. [3.]

3.1 Phone 7 -käyttöliittymän ominaisuudet ja käsitteet

Kun Phone 7 käyttöliittymää alettiin suunnitella, päätettiin ensiksi hylätä aiempien Mobile-käyttöjärjestelmien pyrkimys tarjota käyttäjälle samankaltainen käyttökokemus kuin Microsoftin PC-käyttöjärjestelmissä. Phone 7 -käyttöliittymä ei muistutakaan lainkaan samaa versionumeroa kantavaa Windows 7 -käyttöjärjestelmän käyttöliittymää.

Phone 7 päänäkymä koostuu ns. *Tile:sta* – tiilistä, joiden painaminen käynnistää tiiliä esittävän sovelluksen. Tiilit voivat myös esittää dynaamisesti vaihtuvaa informaatiota. Käyttäjä voi itse muunnella päänäkymän tiilien järjestyksen sekä lisätä tai poistaa tiiliä. Tiilien värin voi vaihtaa muuttamalla käyttöjärjestelmän aksenttiväriä järjestelmän asetuksista.

Kuva 1. Windows Phone 7 päänäkymä koostuu dynaamisesti sisältöään muuttavista tiilistä (Tiles)

Phone 7 järjestelee puhelimen muistissa ja pilvipalveluista tulevan informaation ns. *sisältöhubeihin,* joiden välillä käyttäjä pystyy saumattomasti navigoimaan. Tämä onkin yksi suurista eroavaisuuksista, minkä Phone 7 tekee muihin mobiilikäyttöjärjestelmiin verrattuna. Phone 7 suunnittelussa pyrittiin rikkomaan sovellusten väliset rajat ja tarjota käyttäjälle yhtenäinen käyttökokemus, jossa kaikki tarvittava tieto on korkeintaan muutaman painalluksen takana, ilman että käyttäjän tarvitsee jokaista käyttötapausta varten käynnistää ko. käyttötapauksen suorittava sovellus.

3.2 Käyttöjärjestelmän sovellussuoritin ja sovelluksen elinkaari

Windows Phone 7 poikkeaa Androidin ja iOS sovellussuorituksessa siten, että Phone 7 ei salli kolmansien osapuolien sovellusten taustalla ajamista. Kun käyttäjä navigoi pois sovelluksesta, niin sovellus ei jää pyörimään käyttöjärjestelmän taustalle, vaan käyttöjärjestelmä sulkee ohjelman kokonaan. Phone 7 tarjoaa sovellukselle kuitenkin mahdollisuuden tallentaa sulkemishetkeä edeltävän tilansa, josta sovelluksen suoritusta jatketaan, kun käyttäjä taas palaa käyttämään sovellusta. Tämä ns. *Tombstoning*-tekniikka

konkretisoituu siten, että käyttäjä ei välttämättä edes huomaa, että sovellus on ollut kokonaan pois päältä sillä välin, kun käyttäjä on käyttänyt puhelimen muita ominaisuuksia.

Sovelluksen tilan tallennus on kuitenkin kokonaan sovelluskehittäjän käsissä, eikä käyttöjärjestelmä itsessään suorita Tombstoningia kolmansien osapuolien sovelluksille. Sovelluksen voi kuitenkin jättää käyntiin puhelimen lukitustilan ajaksi, mutta silloin kehittäjän on huolehdittava, että sovellus ei kuluta kohtuuttoman paljon laitteen akkuvarausta.

Kuva 2. Phone 7 -sovelluksen elinkaari. Kuva perustuu MSDN:n artikkelissa Designing Applications for Windows Phone esiintyvään kaavioon. [14]

3.3 Laitevaatimukset

Vanhojen mobiilikäyttöjärjestelmien sovelluskehityksen haasteellisin puoli oli identtisen käyttökokemuksen tarjoaminen päätelaitteiden loppumattomassa kirjossa. Kehittäjä ei voinut tehdä minkäänlaisia olettamuksia päätelaitteen näyttökoosta, prosessorin suorituskyvystä, käytössä olevan muistin määrästä tai vaikka päätelaitteesta löytyvien lisäkomponenttien, kuten kompassin, gps-paikantimen tai kiihtyvyysanturin, olemassaolosta. Sovelluskehittäjän oli siis usein pakko, samaa käyttöjärjestelmää hyödyntävienkin laitteiden välillä, kohdentaa sovellus vain tiettyyn osaan päätelaitteista. Tämä tilanne oli myös Windows Mobile -sovelluskehityksessä.

Windows Phone 7 sovelluskehitystä helpottamaan määritettiin Phone 7 -laitteille tarkat minimivaatimukset päätelaitteen ominaisuuksille sekä tiukkaan standardisoitu laitealusta. Vaatimusten mukaan kaikista Windows Phone 7 -laitteista tulee löytyä seuraavat ominaisuudet:

- Kapasitiivinen neljä pistettä tunnistava WVGA-monikosketusnäyttö (480x800)
- Vähintään 1 GHz ARM v7 ”Cortex/Scorpion” -suoritin
- DirectX 9 -näytönohjain
- 256 MB RAM-muistia ja 8GB Flash-muistia
- Kiihtyvyysanturi kompassilla, valotunnistin, etäisyystunnistin ja A-GPS
- 5 megapikselin kamera LED-salamalla
- FM-radiovastaanotin
- 6 fyysistä näppäintä (back, start, search, camera, power/sleep, volume Up/volume Down).

Cortex- ja Scorpionsuorittimilla viitataan ARM Cortex A8 -suorittimeen (iPhone 3GS) sekä Qualcomm Scorpioniin (HTC HD2, Google Nexus One). [3.]

Windows Phone 7 -laitteen tiukat minimivaatimukset pudottavat ainakin tällä hetkellä tyystin pois halvemmille markkinoille kohdennetut Phone 7 -laitteet. Komponenttien hinnat ovat kuitenkin koko ajan laskussa, joten halvempiakin Phone 7 -laitteita saatetaan nähdä jo piakkoin.

4 Phone Marketplace

Jokaisessa modernissa mobiilikäyttöjärjestelmässä on nykyisin käyttöjärjestelmään tiukasti integroitu kauppapaikka, jossa sovelluskehittäjät voivat helposti levittää sovelluksiaan. Toisaalta kuluttajan on vaivatonta laajentaa päätelaitteensa ominaisuuksia kauppapaikalta löytyvillä lisäsovelluksilla. Mobiilikäyttöjärjestelmäsodan painopiste onkin siirtynyt vahvasti käyttöjärjestelmän ympärille kehittyvän sovellusekosysteemin elinvoimaisuuteen. Tämän sodan voittajiksi selviytyvät ne käyttöjärjestelmät, jotka pystyvät houkuttelemaan eniten sovelluskehittäjiä taakseen tekemään käyttöjärjestelmän päälle rikkaita, laadukkaita, hauskoja ja hyödyllisiä sovelluksia ja pelejä.

Phone Marketplace on Microsoftin ylläpitämä kauppapaikka Windows Phone 7 -sovellusten jakelemiseen. Kauppapaikalle kuka tahansa Phone 7 -kehittäjäksi rekisteröitynyt yritys, tai yksityishenkilö, voi lähettää omia sovelluksiaan. Sovellusten pitää kuitenkin ennen kauppapaikalle pääsemistä läpäistä Microsoftin sertifiointiprosessi, jossa sovelluksen luotettavuus ja sisällön sopivuus testataan.

Kuva 3. Phone 7 Marketplace -käyttöliittymä.

4.1 Phone 7 -sovelluskehittäjäksi rekisteröityminen

Sovelluskehittäjän rekisteröitymisprosessissa Microsoft pyrkii varmistamaan kehittäjän oikean henkilöllisyyden välttyäkseen haittaohjelmatehtailijoilta. Tämän takia kehittäjän pitää varautua lähettämään sähköpostitse skannattuja henkilöpapereita ja käsin tehtyjä allekirjoituksia.

Rekisteröityminen maksaa tavalliselle kehittäjälle vuotuisesti 99 dollaria. Jos kehittäjä on opiskelijana jossain Microsoftin tunnistamista koululaitoksista, rekisteröityminen ei maksa mitään.

Koko rekisteröitymisprosessi kestää yleensä muutaman viikon. Samalla, kun rekisteröityminen on suoritettu, aukeaa kehittäjälle myös mahdollisuus avata enimmillään 5 kpl Windows Phone -päätelaitteita, joissa kehittäjä voi testata sovellustaan. Jotta tämä mahdollisuus aukeaisi, pitää kehittäjän ainakin tällä hetkellä, outoa kyllä, lähettää ensin yksi sovelluksensa testattavaksi Microsoftille, ennen kuin rekisteröitymisprosessi siirtyy seuraavaan vaiheeseen ja mahdollisuus viedä omia sovelluksiaan Windows Phone -päätelaitteille aukeaa. Tähän riittää kuitenkin, että kehittäjä lähettää testattavaksi millaisen tahansa sovelluksen, eikä sovelluksen tarvitse läpäistä Microsoftin sertifiointiprosessia.

4.2 Sovelluksen sertifiointiprosessi

Ennen kuin kehittäjä voi julkaista oman sovelluksensa Market Placeen, pitää sovelluksen käydä läpi Microsoftin määrittelemät laatutestit, joilla pyritään takaamaan se, että sovellus on riittävän vakaa ja turvallinen, eikä kuluta kohtuuttomasti akkua tai häiritse puhelimen muiden ominaisuuksien käyttöä, kun sovellus on päällä. Testeissä pyritään myös varmistamaan, että sovelluksen sisältö ei ole loukkaavaa tai muuten sopimatonta.

Kuva 4. Marketplace julkaisuprosessi. [15.]

Microsoft takaa, että sertifiointitulokset toimitetaan aina viiden työpäivän kuluessa, joskin jotkut kehittäjät ovat raportoineet joutuneensa odottelemaan testituloksia jopa kymmenen päivää.

Kaikki sertifioinnin edellyttämät pykälät ovat dokumentoitu ja avointa tietoa kaikille kehittäjille. Osa pykälistä onkin kerännyt jo paljon kritiikkiä ja kummastusta, lähinnä niiden asettamien rajoitteiden vuoksi. Esim. kriteereissä mainitaan mm. tarkat määrät, miten paljon sovellus saa maksimissaan kuluttaa akkua puhelimen ollessa lukittuna sekä kuinka kauan sovelluksen avautuminen responsiiviseen tilaan saa maksimissaan kestää.

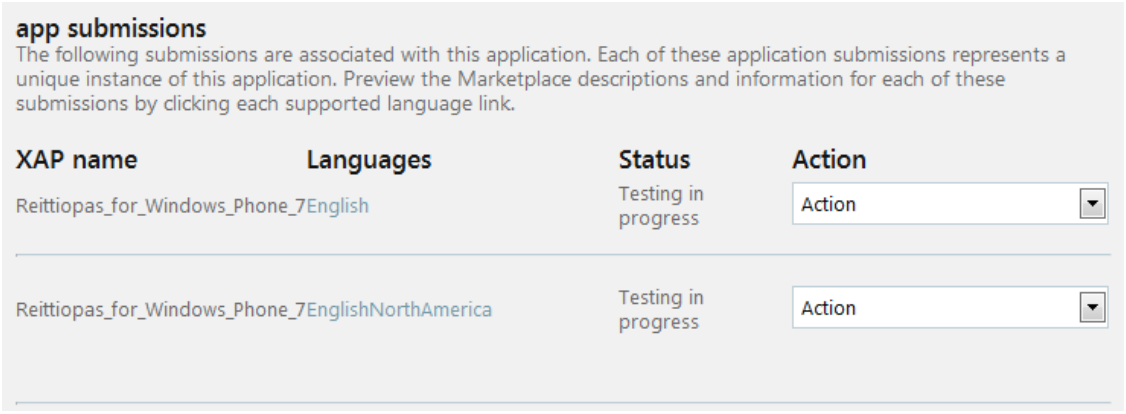
Osa kriteereistä on tehty myös suojelemaan käyttäjän identiteettiä. Esim. sovellukset, jotka hyödyntävät Windows Phone 7:n sijaintitietorajapintoja, pitää kysyä käyttäjältä erikseen lupa hyödyntää näitä tietoja, ja käyttäjälle tulee tarjota mahdollisuus kytkeä tietojen käyttö pois päältä sovelluksessa.

Jokaisessa Phone 7 -laitteessa olevan Back-painikkeen odotetaan myös toimivan kaikkien sovelluksien sisällä taaksepäinnavigointipainikkeena, ja tämän toiminnon ylikirjoit-

taminen omassa sovelluksessa, vaikkakin tehty mahdolliseksi, aiheuttaa automaattisen hylkäyksen testiprosessissa. Kehittäjä voi tietenkin edelleen kytkeytyä back-painikkeen nostamaan tapahtumaan ja suorittaa toimenpiteitä, tai vaikka peruuttaa koko painalluksen käsittelyn hetkeksi, kunhan lopputulos on kuitenkin se, että sovellus navigoi back-painikkeen painalluksesta edelliselle auki olleelle sivulle, tai jos ollaan sovelluksen etusivulla, niin ohjelman pitäisi sulkeutua kokonaan.

4.3 Sovelluksen lokalisointi useille kielialueille Market Placessa

Koska oman sovelluksen voi Market Placessa jakaa maailmanlaajuisesti, voi olla hyvä idea myös kääntää ohjelmasta useita kieliversioita. Market Place tukeekin tätä ominaisuutta oikein hyvin, ja omasta sovelluksesta voi lähettää useita sovelluspaketteja, jotka voivat sisältää eri kielitiedostot. Nämä sovelluspaketit voidaan myös toimittaa eri metatiedoilla, jotka sisältävät ohjelman myyntipuheet ja mainoskuvat.



Kuva 5. Market Placen sovelluspakettien statusnäkymä, jossa kaksi eri kieliversiota testauksessa.

Tällä hetkellä saatavilla olevat kielialueet Market Placessa ovat

- Englanti (yleismaailmallinen)
- Englanti
- Ranska
- Saksa
- Italia
- Espanja

Näistä kullekin kielialueelle voi lähettää yhden sovelluspaketin metatietoineen. Kukin kieliversio joutuu kuitenkin harmillisesti käymään läpi saman sertifiointiprosessin, eikä prosessi nopeudu yhtään, vaikka aiempi kieliversiopaketti olisikin jo läpäissyt sertifiointitestit.

5 Windows Phone -sovelluskehitys

Windows Phone 7 tarjoaa sovellusalustana Windows-kehittäjille tutun .NET-virtuaaliympäristön, jossa kehittäjä voi hyödyntää sovelluksissaan kahta erityyppistä sovellusalustaa. Näistä Silverlightia voidaan hyödyntää rikkaiden liiketoiminta- ja mediasovellusten kehittämiseen, ja XNA:ta alustana mobiilipeleille. Koska alusta on tiukasti standardisoitu, voi kehittäjä olla melko varma, että saadessaan ohjelman toimimaan yhdellä Phone 7 -laitteella, sen toiminta on identtistä myös muissa laitteissa.

5.1 Silverlight for Windows Phone

Silverlight toimi alun perin ainoastaan Adobe Flashin tapaisena html-sivun sekaan upotettuna alustana rikkaille web-sovelluksille, mutta Phone 7 myötä alusta irroitettiin toimimaan osana Phone 7 -käyttöliittymäteknologiaa.

Silverlight-käyttöliittymän vektorit ja animaatiot kuvataan XAML (Extensible Application Markup Language) -kielellä, jotka päivittävät taustalla olevan luokkakokoelman tilaa. Silverlight siis käyttää piirtotekniikkanaan ns. *retain-moodia,* jossa luokkakokoelma pitää sisällään koko sovelluksen graafiset piirto-ohjeet, jotka Silverlight-piirtomoottori voi sitten piirtää kerralla, optimaalisella tavalla, näytölle.

```
<!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION"
                       Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0"
                       Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
    </Grid>
```

Esimerkki 1. Tyypillinen Silverlight for Windows Phone XAML-merkkikielellä kuvattu käyttöliittymänäkymä

Silverlight for Windows Phone ei kuitenkaan tarjoa koko .NET- ja Silverlight for Windows -palettia, vaan osa näiden alustojen toiminnallisuuksista on jätetty Silverlight for Windows Phonen ulkopuolelle. [7.] Eroavaisuudet Silverlightin ja Silverlight for Windows Phonen välillä onkin hyvä olla ohjelmistokehittäjän tiedossa, jos kehittäjä on aiemmin tottunut käyttämään Silverlightia.

5.2 XNA Game Studio 4.0

Mobiilipelit ovat kasvaneet älypuhelimissa erääksi tärkeimmistä kuluttajia houkuttelevista elementeistä, ja suosituimmat mobiilipelit ovat kasvaneet hetkessä todellisiksi maailmanlaajuisiksi hiteiksi. Hittipelin syntyminen mobiilikäyttöjärjestelmän päälle houkuttelee käyttöjärjestelmälle uusia käyttäjiä, ja voidaankin varovasti arvioida, että juuri peliteollisuus on suurin vaikutin mobiilikäyttöjärjestelmien välisessä ekosysteemisodassa.

Silverlight for Windows Phone on oivallinen vastaus rikkaiden liiketoiminta- ja mediasovellusten luontiin, mutta se ei juurikaan sovellu alustaksi moderneille peleille. Siksi Windows Phone 7 tarjoaa pelikehittäjille erillisen XNA Game Studio -ohjelmointiympäristön, joka on jo aiemmin toiminut työvälineenä XBox- ja PC-

pelikehittäjille. XNA hyödyntää piirtomoottorinaan Microsoft DirectX -rajapintaa, mutta se kätkee omilla DirectX-kääreluokillaan DirectX:n monimutkaiset toiminnallisuudet alleen, ja suoraviivaistaa näin rajapinnan käyttöä sovelluskoodissa.

XNA sisältää myös valmiina suuren osan tyypillisimmistä pelikoodielementeistä, kuten pelisilmukan (game loop) ja kattavan pelimatematiikkakirjaston, joka sisältää mm. apuluokat vektori/matriisilaskentaa varten. Myös ns. Content Pipeline, eli sisältökanava, jolla peliin voidaan viedä esim. 3d-malleja ja kuvatiedostoja, löytyy XNA-kehikosta valmiina.

XNA, kuten Silverlight, tarjoaa kehittäjälle vain osan tekniikasta. Esim. räätälöityjen HLSL (High Level Shader Language) -varjostimien käyttö ei ole mahdollista XNA for Windows Phonessa. Kehittäjän on myös huomioitava hyvin rajattu käytössä oleva laitteen suorituskyky verrattuna XBox- tai PC-pelikehitykseen. Toisaalta taas XNA for Windows Phone tuo mukanaan joukon uusia, mm. kosketusnäytön ominaisuuksien hyödyntämistä tukevia, luokkia ja rajapintoja. [4.]

5.3 Sovelluksen kohdentaminen useille Windows Phone 7 -päätelaitteille

Koska Phone 7 -laitteiden ominaisuusvaatimukset ovat poikkeuksellisen tarkoin määritelty, on sovelluskehittäjän helppo varmistaa sovelluksensa mahdollisimman laadukas toimivuus kaikissa Phone 7 -laitteissa. Suurin huomioitava tekijä sovelluksen käyttöliittymää toteutettaessa on laitteiden erot näyttökoissa ja -tarkkuuksissa. Jos tätä seikkaa ei oteta huomioon, niin käyttöliittymän tekstit/muut visuaaliset elementit saattavat jäädä liian pieniksi ihmissilmälle vaivattomasti havaittaviksi.

Kuva 6. Näyttökokovertailu, LG Optimus Q (vas.) ja HTC HD7. Laitteet ovat kuvissa keskenään oikeassa mittakaavassa.

Tätä raporttia kirjoitettaessa pienin Phone 7 -laitteessa oleva näyttö löytyi LG Optimus 7 Q (3.5”) ja suurin HTC HD7 (4.3”). [11]

Suunnitellessa Phone 7 -sovelluksen visuaalista kerrosta ei suunnittelijan tarvitse kuitenkaan huomioida erilaisia näyttöresoluutioita, sillä kaikki Phone 7 -laitteet käyttävät resoluutionaan 800x480 kuvapisteen tilaa. Sen sijaan suunnittelijan kannattaa ottaa huomioon käyttöjärjestelmän väriteema-asetukset, joilla käyttäjä voi valita mieleisensä aksentti- ja taustavärin. On hyvän tyylin mukaista, että oman sovelluksen visuaalinen kerros muokkautuu dynaamisesti käyttäjän valitsemaan väriteemaan, ja silloin on testattava huolellisesti, että kaikki visuaaliset elementit näkyvät kaikissa mahdollisissa väriasetuksissa. Tämän haasteen voi kuitenkin helposti välttää pitäytymällä sovelluksen suunnittelussa Microsoftin valmiissa väri- ja tyylimäärittelyissä, joihin suunnittelijallakin on pääsy ja joissa on jo otettu tämä seikka huomioon.

5.4 Metro-design -käyttöliittymäilmeen suuntaviivat

Windows Phone 7 käyttöliittymä, työnimeltään *Metro,* jäljittelee nimensä mukaisesti (Yhdysvaltain) suurkaupunkien metroasemien sekä lentokenttien opasteiden muotokieltä. Lisäksi käyttöliittymän filosofian pohjaksi otettiin seuraavat viisi periaatetta [12.]:

1. siisti, kevyt, avoin, nopea.
2. sisältö ennen kikkailuja
3. laitteen ja ohjelmiston yhteistyö
4. ensiluokkainen liike
5. sielukkuus ja eloisuus.

Periaatteiden ensimmäisellä ja toisella kohdalla tarkoitetaan sitä, että Phone 7 -käyttöliittymän tulisi erottua muista mobiilikäyttöliittymistä nostamalla typografian käyttöliittymän pääelementiksi sen sijaan, että käytettäisiin paljon muotoja, ikoneita ja kuvia. Näin sisällölle, joka on loppukäyttäjän kannalta relevanttia, jää enemmän tilaa, ja käyttöliittymä on myös helpommin lähestyttävissä uudelle käyttäjälle.

Kuva 7. Phone 7 *People Hub.* Käyttöliitymän pääelementiksi nousee tekstin, eli typografian käyttö sekä esitystavan yksinkertaisuus ja keveys.

Kolmas periaate viittaa siihen, että päätelaitteen ja sen päällä pyörivän käyttöjärjestelmän ja ohjelmiston on tehtävä saumatonta yhteistyötä parhaan käyttökokemuksen saavuttamiseksi.

Neljännellä periaatteella tarkoitetaan sitä, että käyttöliittymän kosketus- ja sormieleiden tulee vastata Windows 7 -PC-käyttöjärjestelmän tukemia kosketusominaisuuksia, ja käyttökokemusta parantamaan käytetään laitekiihdytettyjä liike- ja siirtymisanimaatioita.

Viimeinen periaate nostaa tärkeään rooliin käyttöliittymän mukauttamista käyttäjän henkilökohtaisiin mieltymyksiin sekä käyttöliittymäsisällön jatkuvaa automaattista päivittymistä nykyhetkeen ja tilanteeseen.

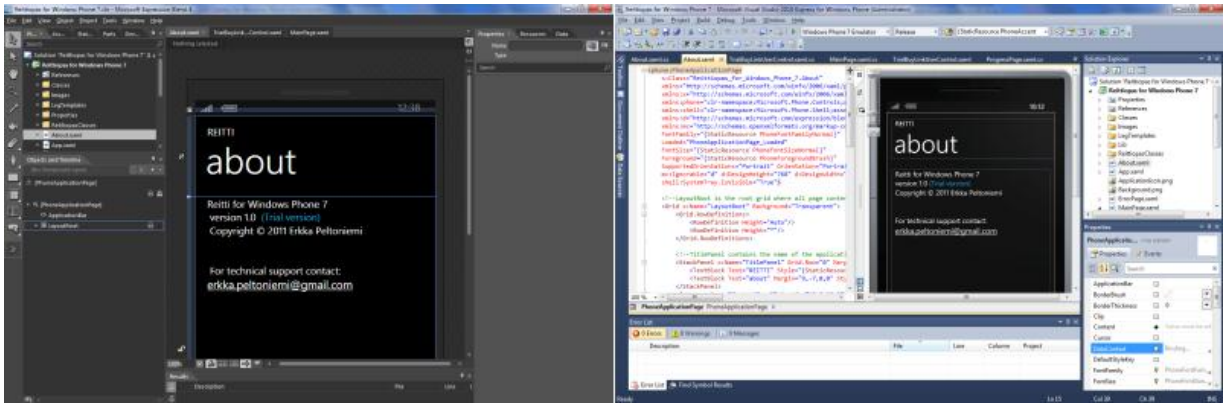
5.5 Kehitystyökalut

Windows Phone 7 -sovelluskehittämiseen Microsoft tarjoaa kehittäjille ilmaiset työkalut. Microsoft Visual Studio Express for Windows Phone on käytännössä karsittu versio yhtiön maksullisesta Visual Studio 2010 -kehitysympäristöstä, joka sisältää kuitenkin kaikki oleellisimmat toiminnallisuudet joilla Phone 7 -sovelluskehitystä voi tehdä. Kehitysympäristöstä löytyy myös integroitu Phone 7 -emulaattori, jolla omaa ohjelmaa voi testata, jos käytössä ei ole oikeaa päätelaitetta. Emulaattorilla voi kokeilla ohjelman ajamista erilaisissa puhelimen asennoissa, jos ohjelmaan halutaan rakentaa tuki sekä pysty- että vaakasuuntaisille tiloille.

Kuva 8. Windows Phone 7 -emulaattori

Microsoft Visual Studio Express for Windows Phonen suurin puutos on versionhallintaintegraatiotyökalujen puute. Tämä voi ohjelmistokehittäjälle olla kivuliasta, jos kehittäjä on tottunut käyttämään Visual Studion integroituja TFS (Team Foundation Server) -työkaluja.

Visual Studion lisäksi Phone 7 -sovelluskehitystä voidaan tehdä myös Microsoft Expression Blend 4 -kehitysympäristöllä, joka on enemmän painottunut ohjelman visuaalisen kerroksen luontiin. Expression Blend 4 on maksullinen sovellus, joka tarjoaa suunnittelijalle visuaaliset työkalut myös animaatioiden luontiin sekä vektorikäyttöliittymän piirtämiseen. Jotta Windows Phone 7 -sovelluskehitys olisi mahdollisimman intuitiivista, kannattaa kehittäjän totutella käyttämään tehokkaasti molempia kehitysympäristöjä.



Kuva 9. Sama projekti ja projektitiedosto avattuna Microsoft Expression Blend 4 (vas.) ja Microsoft Visual Studio Express for Windows Phone -kehitysympäristöissä.

6 Reitti 1.0 WP7-sovelluksen kehittäminen Windows Phone -käyttöjärjestelmälle

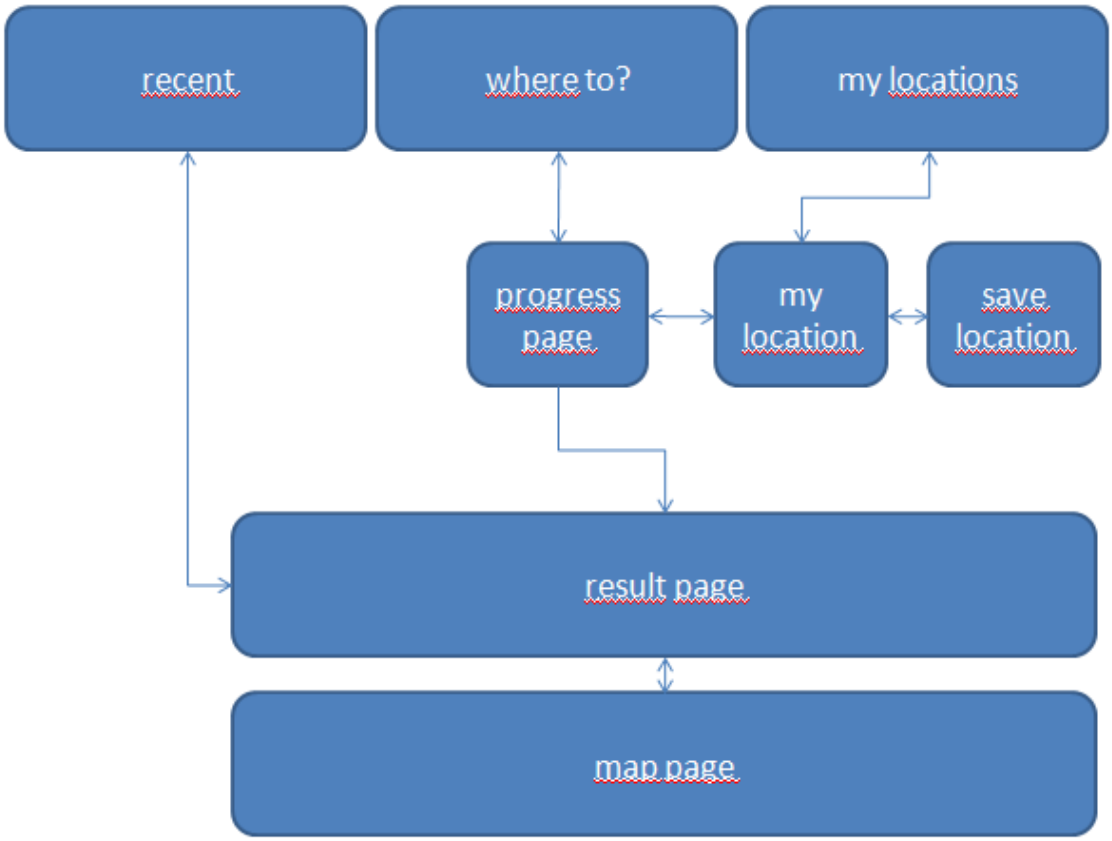
Tässä luvussa käydään läpi Market Placeen julkaistun Windows Phone 7 -sovelluksen mielenkiintoisia osakokonaisuuksia ja esitellään myös pikaisesti Helsingin seudun liikenteen Reittiopas API, jota sovelluksessa hyödynnetään. Reitti käyttää teknisessä toteutuksessaan laajan määrän Silverlight for Windows Phonen tarjoamia ominaisuuksia, joita lähestytään koodiesimerkein pohtien samalla hyviä toteutuskäytäntöjä.

6.1 Yleiskuvaus Reitti-sovelluksen palveluista ja arkkitehtuurista

Reitti-sovelluksessa pyritään yhdistämään HSL:n Reittiopas API:n toiminnallisuudet (Reittiopas API esitelty tarkemmin kappaleessa 6.2) ja Windows Phone 7 -laitteen tarjoamat palvelut toimivaksi ja tiiviiksi kokonaisuudeksi. Heti sovellusta avattaessa Reitti alkaa paikantaa käyttäjän sijaintia Windows Phone Location Services -rajapintojen kautta. Kun sijainti on löytynyt, voi käyttäjä alkaa hakemaan reittejä nykysijainnista mihin tahansa HSL:n julkisen liikenteen alueella olevaan sijaintiin. Reitti tuo käyttäjälle kolme parasta reittiä vieritettäviksi listanäkymiksi, joissa on kuvattu symbolein ja tekstin avulla reitillä käytettävät julkisen liikenteen linjat ja niiden aikataulut. Toisaalta käyttäjä voi myös tarkastella yksittäistä reittiä karttanäkymällä, johon reitti piirretään.

Käyttäjä voi myös tallentaa mielipaikkojaan puhelimen muistiin sekä tarkastella viimeisimpien suoritettujen reittihakujen tuloksia.

Reitti-sovelluksen käyttöliittymä on jaettu viiteen eri näkymään, jotka Silverlight for Windows Phone -alustassa periytyvät *Microsoft.Phone.Controls.PhoneApplicationPage* -luokasta. Sovelluksen päänäkymä jakautuu vielä kolmeen erilliseen alinäkymään (recent, where to? ja my locations), joiden välillä käyttäjä navigoi sivuttaispyyhkäisyliikkeellä. Näkymät ja niiden väliset navigaatiorelaatiot on esitetty kuvassa 10.



Kuva 10. Sovelluksen näkymät ja navigaatiopolut

Silverlight for Windows Phone -teknologiassa näkymän toteutus jaetaan käyttöliittymäkomponentit ja niiden asettelutiedot sisältävään XAML-tiedostoon sekä sitä vastaavan sovelluslogiikan sisältävään c#-kooditiedostoon. Reitin perusarkkitehtuuri on sovelluksen tiiviydestä johtuen valittu hyvin kevyeksi, eikä sovelluksen teknisessä toteutuksessa ole tämän takia lähdetty seuraamaan mitään suosituista kerrosarkkitehtuureista. Kaikki toiminnallinen sovelluslogiikka jakautuukin tismalleen niitä vastaavien käyttöliittymänäkymien mukaan. Näkymät ja niiden tarkoitukset on kuvattu tarkemmin Taulukossa 2.

Näkymän nimi	Näkymän tarkoitus
recent	Listanäkymä viidestä viimeisimmästä hakutuloksesta.
where to?	Hakunäkymä, joka sisältää hakutekstilaatikon, sekä näyttää tietoa GPS-paikannuksen tilasta.
progress page	Näkymä, jossa näytetään liikkuvaa latausanimaatiota pitkien asynkronisten operaatioiden aikana.
my locations	Näkymässä esitetään 6 kpl käyttäjän tallentamia sijainteja.
my location	Näkymä esittää yhden käyttäjän tallentaman sijainnin tiedot.
save location	Näkymä toimii tallennetun sijainnin tietojen syöttölomakkeena.
result page	Hakutulossivu, jossa esitetään kolmen sopivimman reitin tiedot vertikaalisina symbolilistoina.
map page	Näkymä, jossa voidaan tarkastella yksittäistä reittiä karttanäkymänä.

Taulukko 1. Kuvaus Reitti-sovelluksen näkymistä.

6.2 Reittiopas API pääpiirteissään

Helsingin seudun liikenne (HSL) tarjoaa kaikesta julkiseen ja kevyeen liikenteeseen keräämästään tiedosta avointa tietorajapintaa, jolla sovelluskehittäjä voi hakea tietovarastosta monipuolista informaatiota aikatauluista, pysäkeistä, kevyen liikenteen kulkuväylistä ja reitityksestä. Rajapinta on monipuolisuudestaan huolimatta suoraviivainen, ja sen ympärille onkin syntynyt kasvava ekosysteemi sovelluksia lukuisille alustoille, joiden päätarkoitus on helpottaa ihmisten liikkumista pääkaupunkiseudulla.

HSL tarjoaa verkossa suosittua html-pohjaista verkkopalvelua (http://www.reittiopas.fi) jossa käyttäjälle palautetaan julkisia liikennevälineitä hyödyntäviä reittiohjeita, kun käyttäjä on määrittänyt lähtö- ja päätöspisteen sekä muita valinnaisia parametreja.

Tämä palvelu ei kuitenkaan ole kovin käyttäjäystävällinen, kun sitä käytetään nykyaikaisella kosketusnäytöllisellä älypuhelimella. Tämä puute, sekä toisaalta avoin Reittiopas-rajapinta, onkin avannut mobiilikehittäjille mahdollisuuden toteuttaa Reittiopas-rajapintaa hyödyntäviä asiakassovelluksia mobiililaitteisiin ja muille alustoille. Tällaisia sovelluksia löytyy ainakin Applen iOS- ja Googlen Android -mobiilikäyttöjärjestelmille. Nämä sovellukset osaavat monipuolisesti hyödyntää tyypillisiä nykyaikaisesta mobiililaitteesta löytyviä ominaisuuksia, kuten GPS-paikannusta, ja tuovat sitä kautta lisää käytettävyyttä ja monipuolisuutta omiin sovelluksiinsa.

Reittiopas-rajapinnan kanssa kommunikoidaan lähettämällä palvelimelle HTTP-GET-kutsuja, joissa parametrit välittyvät palvelimelle tulkattaviksi http-osoiteparametreina. kun palvelin on muodostanut vastauksen, se palautetaan asiakkaalle. Vastauksen formaatin voi valita parametrein, ja Reittiopas API osaa palauttaa sen joko XML- tai JSON -formaatissa.


```
<?xml version="1.0" encoding="iso-8859-1" ?>
<MTRXML version="1.0">
  <GEOCODE key="tee">
    <LOC name="Teeripuisto" number="" city="Helsinki" code="1701" address="" type="3" category="poi" x="2556686" y="6682815" lon="25.02051" lat="60.2528" />
    <LOC name="Teekkaripolku" number="" city="Espoo" code="" address="" type="900" category="street" x="2546340" y="6675352" lon="24.832" lat="60.18713" />
    <LOC name="Teemuntie" number="" city="Helsinki" code="" address="" type="900" category="street" x="2557985" y="6685213" lon="25.04465" lat="60.27414" />
    <LOC name="Teerenpelinpolku" number="" city="Helsinki" code="" address="" type="900" category="street" x="2556532" y="6682578" lon="25.01767" lat="60.2507" />
    <LOC name="Teerentie" number="" city="Kirkkonummi" code="" address="" type="900" category="street" x="2524959" y="6686629" lon="24.44304" lat="60.2902" />
    <LOC name="Teerikuja" number="" city="Vantaa" code="" address="" type="900" category="street" x="2559094" y="6693721" lon="25.06718" lat="60.35033" />
    <LOC name="Teerikukonkuja" number="" city="Helsinki" code="" address="" type="900" category="street" x="2556861" y="6683030" lon="25.02373" lat="60.25471" />
    <LOC name="Teerikukontie" number="" city="Helsinki" code="" address="" type="900" category="street" x="2556888" y="6682971" lon="25.0242" lat="60.25417" />
    <LOC name="Teerimäentie" number="" city="Kerava" code="" address="" type="900" category="street" x="2560257" y="6698983" lon="25.08981" lat="60.39737" />
    <LOC name="Teerimäentie" number="" city="Vantaa" code="" address="" type="900" category="street" x="2562518" y="6686969" lon="25.12709" lat="60.28923" />
    <LOC name="Teerimäentie" number="" city="Espoo" code="" address="" type="900" category="street" x="2536615" y="6673635" lon="24.65643" lat="60.1727" />
    <LOC name="Teeripolku" number="" city="Vantaa" code="" address="" type="900" category="street" x="2559118" y="6693833" lon="25.06764" lat="60.35133" />
    <LOC name="Teeririnne" number="" city="Vantaa" code="" address="" type="900" category="street" x="2559182" y="6693629" lon="25.06874" lat="60.34949" />
    <LOC name="Teerisuonkuja" number="" city="Helsinki" code="" address="" type="900" category="street" x="2556947" y="6682640" lon="25.02518" lat="60.25119" />
    <LOC name="Teerisuontie" number="" city="Helsinki" code="" address="" type="900" category="street" x="2556822" y="6682723" lon="25.02294" lat="60.25196" />
    <LOC name="Teeritie" number="" city="Vantaa" code="" address="" type="900" category="street" x="2559089" y="6693605" lon="25.06705" lat="60.34929" />
    <LOC name="Teekkarikylä" number="" city="Espoo" code="2222208" address="Otakaari" type="10" category="stop" x="2548445" y="6675512" lon="24.83393" lat="60.18855" />
    <LOC name="Teerisuonkuja" number="" city="Helsinki" code="1382186" address="Teerisuontie" type="10" category="stop" x="2556964" y="6682689" lon="25.02547" lat="60.25091" />
    <LOC name="Teerisuontie" number="" city="Helsinki" code="1382141" address="Teerisuontie" type="10" category="stop" x="2556740" y="6682861" lon="25.0215" lat="60.25321" />
    <LOC name="Teeritie" number="" city="Vantaa" code="4810204" address="Korsontie" type="10" category="stop" x="2559002" y="6694007" lon="25.06559" lat="60.35291" />
  </GEOCODE>
</MTRXML>
```


Kuva 11. XML-muotoinen Reittiopas API:n palauttama vastaus, jossa asiakas on tehnyt paikannimihaun hakusanalla "tee".

Reittiopas API jakautuu yhdeksään eri moduuliin, joista kukin toteuttaa erilaisen toiminnallisuuden. [13.] Reittiopas API -moduulit ovat:

1. Geocoding – geokoodaus palauttaa paikkojen osoitteita ja koordinaatteja vapaasanahakutermiä vastaan.
2. Reverse geocoding – käänteinen geokoodaus palauttaa tietoa lähellä olevista paikoista koordinaattia vastaan.
3. Stop information – pysäkkitieto palauttaa esteettömyys- linja- ja koordinaattitietoa pysäkistä pysäkin tunnusta vastaan.
4. Stops in area – alueella sijaitsevat pysäkit palauttaa määritetyn karttaneliön sisällä sijaitsevat pysäkit.
5. Line information – linjatiedot palauttaa tarkkaa tietoa yksittäisestä linjasta.
6. Routing between two points – reititys palauttaa reittejä määritettyjen pisteiden välillä. Monipuolinen parametriikka.
7. Cycling route – pyöräilyreitit palauttaa polkupyöräilijöille suunnattuja reittejä
8. Data validity – tiedon validointi. Moduulilla voi tarkistaa millä aikavälillä reittiopas-tietokannassa olevat tiedot ovat voimassa.
9. User statistics – käyttäjätiedot-moduuli palauttaa asiakaskohtaisen reittiopastilin käytetyn- ja enimmäiskapasiteetin.

Huomioitavia seikkoja Reittiopas API:n käytössä on koordinaattijärjestelmän valinta. Oletuksena Reittiopas API palauttaa kansainvälisellä tasolla eksoottista KKJ2-koordinaattitietoa, joka ei ole yhteensopivaa GPS-sirujen WGS84-formaatin kanssa. Jotta Reittiopas API:n tietoja voi hyödyntää yhdessä GPS-sirun tietojen kanssa, on parametrisoitava Reittiopas API palauttamaan samaa WGS84-tyyppistä koordinaattitietoa kuin mitä GPS-sirut käyttävät.

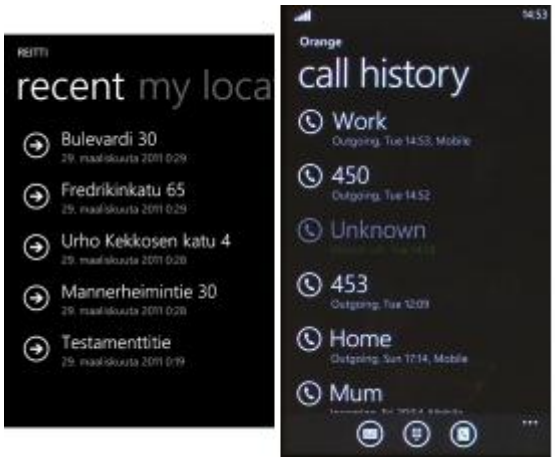
6.3 Reitti-sovelluksen käyttöliittymä

Sovelluksen käyttöliittymää suunnitellessa oli heti selvää, että Reitti-sovelluksen käyttöliittymä pyritään istuttamaan mahdollisimman hyvin Phone 7 -käyttöjärjestelmän käyttöliittymään ja muodostamaan mahdollisimman saumattoman jatkeen käyttöjärjestelmän vakiotoiminnallisuuteen. Silverlight for Windows Phone -

oletuskäyttöliittymäkomponentit tukevat käyttöjärjestelmän Metro-designia pääpiirteissään melko hyvin, mutta jättävät jonkin verran räätälöintityötä myös kehittäjälle. Esimerkiksi käyttöjärjestelmän persoonalliset sivutransitioanimaatiot eivät ole sellaisenaan käytössä kolmannen osapuolen sovelluksissa, vaan animointi pitää tehdä itse.

6.3.1 Reitti-käyttöliittymäkomponentit

Reitti-sovellukseen toteutettiin kloonit Phone 7 -ominaisista pyöreistä ikonipainikkeista, joiden ikonin väritys ja taustaväri muuttuvat painettaessa vastaväreikseen.



Kuva 12. Reitti-sovelluksen recent-näkymä (vas.) ja Phone 7 -käyttöjärjestelmän call history näkymä. Kuvan pyöreät ikonit ovat painikkeita. Reitti pyrkii mahdollisimman pitkälti muistuttamaan Phone 7 -Metrokäyttöliittymää.

Pyöreä ikonipainike voidaan Silverlightilla toteuttaa kokonaan XAML-merkkikielellä tekemällä Silverlight Button -käyttöliittymäkomponentille räätälöity Control Style. Control Style -määritys vastaa XAML:ssa webmaailman CSS-luokkamäärittelyjä. Ikonipainikkeen toteutus on esitetty kokonaisuudessaan liitteessä 1. Kun Control Template on laadittu, on helppoa pudottaa käyttöliittymään uusia ikonipainikkeita:

```
<Button Click="RecentRouteButton_Click"
     Style="{StaticResource EllipseButtonStyle}">
     <ImageBrush ImageSource="Images/appbar.next.rest.png" />
</Button>
```

Esimerkki 2. Control Style käyttö XAML-merkkikielessä, kun on määritelty EllipseButtonStyle-niminen Control Style.

Kuvassa 11 viitataan Silverlightin Button-kontrolliin, jolle määritetään Style-attribuutilla räätälöity tyylimääritys nimeltään EllipseButtonStyle, joka taas on määritelty sovelluksen resurssikirjastoon. Button-komponentille voidaan nyt antaa sisällöksi ImageBrush-elementti, joka viittaa fyysiseen kuvatiedostoon, jonka ikoni halutaan tuoda näytölle. Kuvatiedosto taas toimii ns. läpinäkyvyysmaskina (Opacity mask). Läpinäkyvyysmaski on kuva, jonka avulla voidaan säädellä kuvan alla olevan ui-elementin alfakanavaa. Kun kuvatiedosto sisältää ikonin muodon, voidaan ikonin väriä vaihtaa lennossa vain vaihtamalla alla olevan elementin taustaväriä. Tämä on erityisen hyvä tekniikka kun halutaan varmistaa, että komponentti näkyy oikein, kun käyttäjä on valinnut käyttöön kumman tahansa WP7:n mustasta tai valkoisesta teemaväristä.

6.3.2 Reitti-sovelluksen värimaailma ja dynaaminen reagointi käyttöjärjestelmän väriasetuksiin

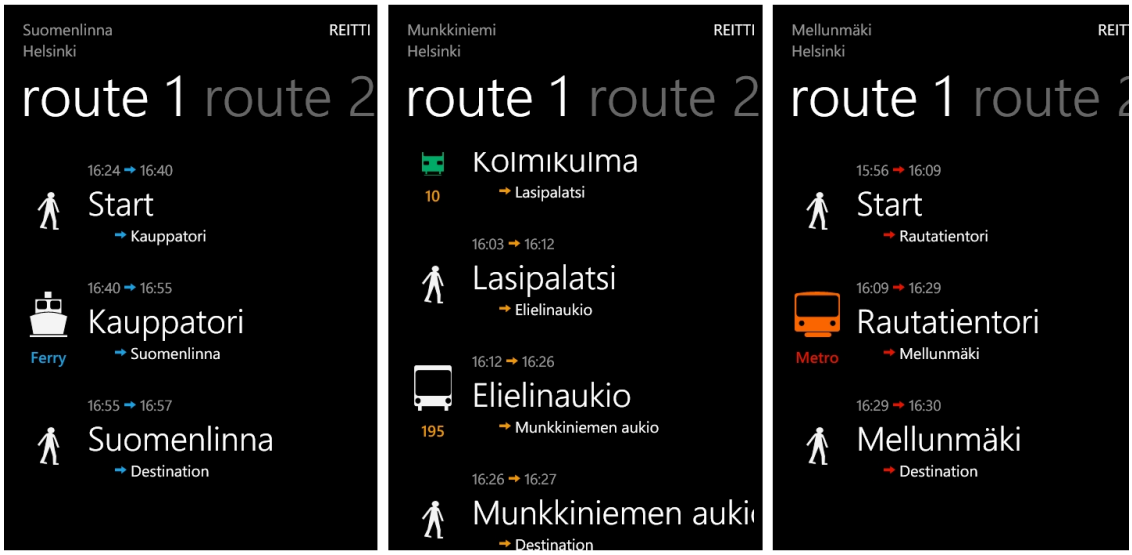
Windows Phone 7 -käyttöjärjestelmä antaa käyttäjän vapaasti valita laitteensa taustavärin dark- tai light-väriteemojen välillä. Tämän lisäksi käyttäjä voi vielä valita ns. aksenttivärin, joka näkyy käyttäjälle esimerkiksi käyttöjärjestelmän päänäkymän Live Tile -taustavärien muutoksella. Näiden väriasetusten yhteisvaikutuksella muodostuu suuri määrä värimaailmoita.

Kuva 13. Windows Phone 7 väriasetukset. Käyttäjä voi valita laitteeseensa käytössä olevan taustavärin (dark/light) sekä aksenttivärin (10 oletusvaihtoehtoa + mahdollinen operaattorispesifi vaihtoehto).

Reitti-sovellus on kehitetty reagoimaan käyttäjän väriasetuksiin. Toteutus tälle on Silverlight for Windows Phonessa yksinkertaista: Kehittäjä voi XAML-merkkikielessä tehdä viittauksia puhelimen resursseihin, ja jos kehittäjä pidättäytyy käyttämään ainoastaan näitä oletusväriresursseja, fontteja ja tekstityylejä, tulee sovelluksesta automaattisesti kaikkia väriasetusyhdistelmiä tukeva sovellus, jonka mikään käyttöliittymäkomponentti ei jää millään yhdistelmällä heikosti näkyväksi tai kokonaan näkymättömäksi.

```
<SolidColorBrush Color="{StaticResource PhoneAccentColor}" />
<SolidColorBrush Color="{StaticResource PhoneBackgroundColor}" />
```

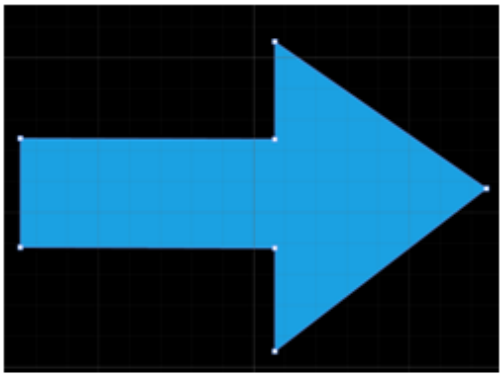
Esimerkki 3. Koodissa määritetään kaksi SolidColorBrush-elementtiä. Ylemmän Color-attribuutti viittaa käyttöjärjestelmän aksenttiväriin ja alempi taustaväriin. Nyt elementit reagoivat dynaamisesti laitteen väriasetuksiin.



Kuva 14. Reitti-sovelluksen tulosnäkymä ja dynaaminen reagointi käyttöjärjestelmän väriasetuksiin. Kuvissa sininen (vas.), oranssi ja punainen aksenttiväri tummalla-taustavärillä.

6.3.3 Silverlight -vektorigrafiikan hyödyntäminen

Silverlight-teknologian XAML-merkkikieltä voidaan hyödyntää myös vektoripohjaisen grafiikan piirtämisessä. Silverlight pitää sisällään jonkin verran valmiita olioita helpottamaan vektorigrafiikan määrittämistä. Oleellisimpia näistä ovat *Rectangle* ja *Ellipse* -luokat nelikulmioiden ja ellipsien määrittämiseen sekä *Path*-luokka monimutkaisempien kuvioiden määrittämiseen.



```
<Path Data="M109,1.5 L240.9095,99.646553 L109,208.5 L109,139.5 L-49.672413,138.90517 L-49.672413,66.146553 L109,66.741379 z" >
    <Path.Fill>
        <SolidColorBrush Color="{StaticResource PhoneAccentColor}"/>
    </Path.Fill>
</Path>
```

Esimerkki 4. Expression Blend -kehitysympäristöstä leikattu kuva ohjelman vektorigrafiikka-piirtonäkymästä ja sen generoima XAML-koodi. Path-luokan Data-attribuutti pitää sisällään kuvion määritelmän. Kuvion täyttöväriksi on asetettu laitteen aksenttiväri.

Expression Blend tarjoaa kehittäjälle visuaalisen työkalun myös vektorigrafiikan piirtämiseen. Ohjelmalla onnistuu hyvin ainakin kevyehkö vektorigrafiikan piirtäminen. Vektorigrafiikan käytön etuja ovat rajaton skaalautuvuus ilman kuvatarkkuuden heikentymistä, sekä sen loputon muokattavuus mm. ohjelmakoodissa. Kaikkia vektorigrafiikka-apuluokkia voi kehittäjä animoida ja manipuloida c#- tai XAML-koodin avulla sovelluksen ajon aikana. Vastaava ei ole helppoa kun käytössä on ainoastaan valmiiksi rasteroituja kuvatiedostoja.

6.4 Kuvaus Reitti-sovelluksen hyödyntämistä Phone 7 -palveluista ja rajapinnoista

Reitti hyödyntää useita Phone 7 -käyttöjärjestelmän ominaisuuksia pyrkien näin tuomaan käyttäjälle henkilökohtaisen ja sujuvan käyttökokemuksen. Heti, kun sovellus on avattu, Reitti ryhtyy Phone 7 Location Services -ohjelmointirajapintoja hyväksikäyttäen etsimään GPS-satelliitteja ja paikantamaan käyttäjän sijainti. Kun käyttäjän sijainti on löydetty, voi käyttäjä alkaa suorittamaan reittihakuja paikannetusta sijainnista. Sijainnin WGS84-muotoinen koordinaatti lähetetään Reittiopas API:lle .NET WebClient -luokan avulla HTTP GET -pyyntönä.

Sovellus hyödyntää GPS:n ja WebClientin lisäksi myös puhelimen kameraominaisuutta. Käyttäjä voi ottaa kuvan mieleisestään paikasta ja sen jälkeen reittihaut ko. paikkaan tapahtuvat kuvaa sormella näpäyttämällä.

Reittiopas API:n palauttamaa reittiä voi tulosnäkymän lisäksi tarkastella myös myös karttapohjalla, jossa karttatiedon ja toiminnallisuuden tarjoaa Bing Maps ja Silverlight Map Control.

Sessiosta toiseen säilyvän pitkäaikaisen tiedon Reitti säilöö WP7:n eristettyyn tietovarastoon (Isolated Storage), joka on sovelluskohtainen eikä muista sovelluksilla ole pääsyä toisten sovellusten tietovarastoon.

6.4.1 Ohjelmoijan pääsy GPS-paikkatietoon WP7:ssa

Kaikista Windows Phone 7 -laitteista löytyy standardoidun laitteiston ansiosta A-GPS (Assisted GPS) -siru, jonka tarjoamaan sijaintitietoon kehittäjällä on pääsy *System.Device.Location* -nimiavaruudesta löytyvän *GeoCoordinateWatcher*-luokan tarjoamilla palveluilla.

```
var watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.High);
            watcher.PositionChanged +=
new EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);
            watcher.StatusChanged +=
new EventHandler<GeoPositionStatusChangedEventArgs>(watcher_StatusChanged);
            watcher.Start();
```

Esimerkki 5. GeoCoordinateWatcher -alustuskoodi. Esimerkissä luokka instansioidaan korkealla sijaintitarkkuudella ja kiinnitetään kaksi tapahtumakäsittelijää. PositionChanged -tapahtuma laukeaa, kun sijainti on muuttunut vähintään MovementThreshold-property:n verran

GeoCoordinateWatcher-luokka antaa mahdollisuuden parametrisoida mm. sijaintitiedon halutun tarkkuuden. Tilanteissa, joissa ei välttämättä tarvita eksaktia sijaintitietoa tai halutaan säästää laitteen akkua, voidaan laitteen akun kestoa tarvittaessa nostaa tinkimällä sijaintitiedon tarkkuudesta.

GeoCoordinateWatcher antaa kehittäjälle myös mahdollisuuden määrittää sijaintitiedon muuttumisen kynnysarvon, jolla voidaan säätää PositionChanged-tapahtuman laukeamisintervallia. Tämä mekanismi mahdollistaa niin ikään akun käytön optimoinnin.

```
void watcher_PositionChanged(object sender, GeoPositionChangedEventArgs<GeoCoordinate> e)
{
    double lat = e.Position.Location.Latitude;
    double lon = e.Position.Location.Longitude;
}
```

Esimerkki 6. Esimerkki PositionChanged-tapahtumakäsittelijästä. Esimerkissä tapahtuma-argumenteista kaivetaan sijainnin leveys- ja pituus -arvot erillisiin double-tyyppisiin muuttujiin.

6.4.2 Kommunikaatio Internet-palveluiden kanssa

Reitti tarvitsee toimiakseen kommunikointia verkon yli Reittiopas API:n kanssa, johon sovellus lähettää http-get-pyyntöjä. Tämän lisäksi verkkoliikennettä tapahtuu vielä toisenkin verkkopalvelun, Microsoft Bing Maps -palvelun kanssa, joka on Microsoftin luoma suora kilpailija Googlen suositulle Google Mapsille. Kommunikaatio näiden palveluiden kanssa on hyvin erityyppistä, Reittiopas API:n kanssa kommunikoidaan *System.Net* -nimiavaruudessa sijaitsevan *WebClient*-apuluokan avulla. Bing Maps -kommunikaatio taas tapahtuu kokonaisuudessaan Microsoftin Map -Silverlight-käyttöliittymäkomponentin verhojen takana (esitelty tarkemmin kappaleessa 6.4.3).

```
WebClient client = new WebClient();

client.DownloadStringCompleted +=
    new DownloadStringCompletedEventHandler(client_DownloadStringCompleted);

client.DownloadStringAsync(new Uri("http://www.hs.fi"));
```

Esimerkki 7. Esimerkkikoodi WebClient-luokan alustuksesta. Luokkaan kiinnitetään tapahtumakäsittelijä joka laukeaa kun DownloadStringAsync-komento on suoritettu taustasäikeessä.

```
void client_DownloadStringCompleted
(object sender, DownloadStringCompletedEventArgs e)
{
    if (e.Error == null)
    {
        string result = e.Result;
        // ...
    }
}
```

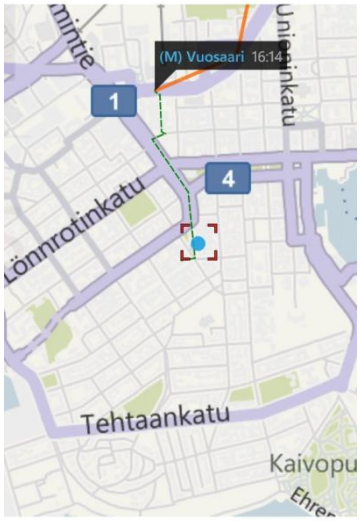
Esimerkki 8. Esimerkkikoodi WebClient DownloadStringCompleted-tapahtumakäsittelijästä. Koodissa tarkistetaan onko pyynnön aikana tapahtunut virheitä esim. tietoliikenteessä.

WebClient-luokkaa voidaan hyödyntää sekä synkronisesti että asynkronisesti. Asynkronisessa tilassa käyttöliittymäsäiettä käytetään vain tapahtumakäsittelijän kiinnittämiseen ja kutsun käynnistämiseen. Tämän jälkeen taustasäie tekee varsinaisen työn ja

työ palautuu lopuksi käyttöliittymäsäikeen tapahtumakäsittelijälle käsiteltäväksi. Koska Silverlightissa käyttöliittymä on kokonaisuudessaan pääsäikeen hallinnassa, on hyvä idea käyttää WebClienttia aina asynkronisesti. Näin käyttöliittymäsäie pystyy jatkamaan käyttäjältä tulevien eleiden ja painallusten käsittelyä pitkien verkon yli lähetettävien pyyntöjenkin aikana.

6.4.3 Bing-karttatietojen käyttö

Microsoft tarjoaa Bing Maps -karttapalvelun tiedot kehittäjien käyttöön Silverlight Map -käyttöliittymäkomponentin kautta. Jotta komponentin voi ottaa käyttöön omassa sovelluksessa, on kehittäjän ensin rekisteröitävä itselleen Bing Maps -kehittäjätili. Kun tili on avattu, kehittäjä voi generoida itselleen sovelluskohtaisen avaimen, joka syötetään Map-kontrollille parametrina. Kontrollin käyttö sovelluksessa on ilmaista silloin, kun sovellus ei käytä Bingin tarjoamaa tieliikennetietoa.



Kuva 15. Reitti-sovelluksen karttanäkymä. Näkymään on piirretty viivoja ja symboleita kuvastamaan reittitietoja

Map-kontrolli antaa kehittäjälle mahdollisuuden piirtää omia symboleita viivoja ja kuvioita karttanäkymän päälle. Keskeisimpiä apuluokkia karttapinnan päälle piirtämiseen ovat *MapPolyline*, jolla voi piirtää useiden koordinaattien kautta kulkevia viivoja, sekä *PushPin*, joka sopii parhaiten kartan tiettyjen kohtien merkitsemiseen.

Käytännössä Silverlight Map -kontrolli jättää aika paljon toivomisen varaa suorituskyvyn ja loppukäyttäjän käytön sujuvuuden suhteen. Käyttökokemus jää tällä hetkellä kauaksi WP7:n mukana tulevasta Maps-sovelluksesta. Komponentti esimerkiksi piirtää karttanäkymää siirrellessä kartan kuvatietoa synkronisesti, jolloin laitteen käyttöliittymäsäie voi olla pitkiäkin aikoja varattuna. Tämä näkyy käyttäjälle ”tökkivänä” kartan vieritys-kokemuksena.

6.5 Microsoft.Phone.Tasks -nimiavaruus ja kameran hyödyntäminen omassa sovelluksessa

Windows Phone 7 -ohjelmointirajapinnat sisältävät pääsyn puhelimen yleisiin toiminnallisuuksiin. Tällaisia toiminnallisuuksia ovat esim. tekstiviesti-, sähköposti- ja puhelinsoittoohjelmien käynnistys, verkkosivun avaaminen selaimeen tai kameraohjelman käynnistäminen. Näille ns. Task- tehtävätyyppisille toiminnallisuuksille on määritetty yhteinen nimiavaruus, *Microsoft.Phone.Tasks.* Kehittäjä voi hyödyntää näitä tehtäväkäynnistimiä omassa ohjelmakoodissaan, ja esim *CameraCaptureTask*:ssa kehittäjä voi kiinnittää tapahtumakäsittelijän Tehtävän *Completed*-tapahtumaan, ja saada näin paluuarvona kuvatiedoston käyttäjän ottamasta kuvasta.

```
CameraCaptureTask t = new CameraCaptureTask();
t.Completed += new EventHandler<PhotoResult>(t_Completed);
t.Show();
```

Esimerkki 9. Esimerkkikoodi kameratehtävälaukaisimen käynnistyksestä.

```
void t_Completed(object sender, PhotoResult e)
{
    if (e.TaskResult != TaskResult.OK) return;
    WriteableBitmap wb = PictureDecoder.DecodeJpeg(e.ChosenPhoto);
}
```

Esimerkki 10. Esimerkkikoodi CameraCaptureTask Completed -tapahtumakäsittelijästä. Koodissa dekoodataan Tehtävän palauttama JPEG-pakattu kuvadata bittikartaksi.

Tehtäväkäynnistäjät tarjoavat kehittäjälle rajatun pääsyn puhelimen ominaisuuksiin. Näiden avulla kehittäjä ei voi esim. rakentaa räätälöityä tekstiviesti- tai kameraohjelmaa. Ne ovat ennemminkin hyvin ennalta käsikirjoitettuja dialogeja laitteen ja käyttäjän välillä, joissa kontrolli siirtyy hetkeksi omasta sovelluksesta käyttöjärjestelmän oletussovelluksille, ja sisältävät sitten jonkinlaisen tapahtuman, jolla oma sovellus voi taas jatkaa dialogin tuloksen kanssa.

6.6 Tiedon varastoiminen puhelimen muistiin

Kaikki Windows Phone 7 -sovellukset ovat rajoitettuja käyttämään tallennustilanaan *Isolated Storage* -tietovarastoa, joka on eristetty alla olevan käyttöjärjestelmän levyjärjestelmästä. Sovelluksilla ei ole myöskään pääsyä muiden sovellusten tietovarastoihin. Sovelluksen kaikki I/O-operaatiot siis rajoittuvat sovelluksen omaan tietovarastoon. Nämä järjestelmärajoitteet on luotu ennen kaikkea tuomaan laitteeseen lisää tietoturvaa. Mikään kolmannen osapuolen sovellus ei pääse käsiksi käyttäjän henkilökohtaisiin tietoihin tai sotkemaan käyttöjärjestelmää ja muita kolmannen osapuolen sovelluksia.

Kaikki Isolated Storage -operaatiot tapahtuvat ohjelmakoodissa *System.IO.IsolatedStorage* -nimiavaruuden *IsolatedStorageFile*-olion kautta.

```
using (var storage = IsolatedStorageFile.GetUserStoreForApplication())
using (var fileStream = storage.CreateFile("test.txt"))
using (var writer = new StreamWriter(fileStream))
{
    writer.Write("Tämä on test.txt -tiedoston testisisältöä...");
}
```

Esimerkki 11. Koodiesimerkissä luodaan test.txt -niminen tiedosto johon kirjoitetaan tekstirivi.

Tiedostojen luku ja kirjoitus tapahtuvat *StreamReader* ja *StreamWriter* -olioiden avulla, joiden konstruktoreihin syötetään viittaus käsiteltävän tiedoston tiedostovirrasta.

```
using (var storage = IsolatedStorageFile.GetUserStoreForApplication())
using (var fileStream = storage.OpenFile("test.txt", FileMode.Open))
using (var reader = new StreamReader(fileStream))
{
    string content = reader.ReadLine();
}
```

Esimerkki 12. Esimerkissä luetaan Esimerkki 11 tallennetun tiedoston ensimmäinen rivi string-muuttujaan.

6.6.1 Kompleksityyppien persistoiminen eristettyyn tietovarastoon binäärisarjoitusmenetelmällä

Usein levyoperaatioissa tulee koodin selkeyden vuoksi halu abstrahoida tallennettava tieto omiin luokkiinsa siten, että ohjelmakoodissa ikään kuin vain ladataan ja tallennetaan olioita levyn ja muistin välillä. Isolated Storagen onkin mahdollista kirjoittaa binäärisarjoitettu tieto suoraan levylle *BinaryWriter*-olion avulla ja lukea se sitten takaisin *BinaryReader*-oliolla.

Esimerkkitoteutus on esitetty Esimerkki 13.3:ssa. Luodaan ensin rajapinta, johon määritellään tarvittavien metodien allekirjoitukset binäärivirtaoperaatioita varten. Seuraavaksi luodaan apuluokka, joka rajapintamäärittelyn avulla osaa lukea ja kirjoittaa rajapinnan toteuttavalle luokan levyluku- ja kirjoitusoperaatiot.

```
public interface ICustomBinarySerializable
{
    void Serialize(BinaryWriter writer);
    void DeSerialize(BinaryReader reader);
}

public static class IsolatedStorageHandler
{
    public static void Save(string filepath,
        ICustomBinarySerializable dataObject)
    {
        using(var storage = IsolatedStorageFile.GetUserStoreForApplication())
        using(var file = storage.OpenFile(filepath,FileMode.OpenOrCreate))
        using(var writer = new BinaryWriter(file))
        {
                dataObject.Serialize(writer);
        }
    }

    public static void Open(string filepath,
        ICustomBinarySerializable dataObject)
    {
        using (var storage = IsolatedStorageFile.GetUserStoreForApplication())
        using (var file = storage.OpenFile(filepath, FileMode.Open))
        using (var reader = new BinaryReader(file))
        {
            dataObject.DeSerialize(reader);
        }
    }
}
```

Esimerkki 13. ICustomSerializable-rajapintakuvaus ja IsolatedStorageHandler-apuluokka.

Nyt kun kaikki binäärioperaatiot on abstrahoitu ja yleistetty, on helppoa kirjoittaa erilaisia luokkia, joihin I/O-operaatiot kohdennetaan. Esimerkkitoteutus on esitetty Esimerkki 14:ssa, jossa *Person*-luokalle on määritetty string- ja int-tyyppiset property:t.

```
public class Person : ICustomBinarySerializable
{
    public int Age { get; set; }
    public string Name { get; set; }

    public void Serialize(BinaryWriter writer)
    {
        writer.Write(Age);
        writer.Write(Name);
    }

    public void DeSerialize(BinaryReader reader)
    {
        Age = reader.ReadInt32();
        Name = reader.ReadString();
    }
}
```

Esimerkki 14. Person-luokka joka toteuttaa ICustomSerializable-rajapinnan

Serialize- ja *DeSerialize*-metodien toteutuksessa on huomattava, että koska luku- ja kirjoitusoperaatiot tapahtuvat bittivirtaan, on myös luokan propertyjen luku- ja kirjoitusjärjestyksen oltava sama.

6.7 Tombstoning-toteutus käytännössä

Kun käyttäjä hyppää pois sovelluksesta, Phone 7 -käyttöjärjestelmä siirtää sovelluksen passiiviseen tilaan. Tombstoning-termillä viitataan sovelluksen tilan tallennukseen ennen kuin sovellus muuttuu passiiviseksi, ja toisaalta tilan palauttamista passiiviseksi asettamista edeltäneeseen tilaan, kun käyttäjä navigoi takaisin sovellukseen.

Windows Phone 7 -kehityksessä tombstoning on kokonaan kehittäjän harteilla, ja on täysin kehittäjästä kiinni, miten täydellisen tilanhallinnan kehittäjä haluaa loppukäyttäjälle tarjota. Suosituksena on, että tombstoning olisi loppukäyttäjälle täysin näkymätön tapahtuma, eikä käyttäjän tarvitsisi edes huomata, että sovellus on käynyt passiivisessa tilassa.

Tombstoningin kaksi oleellista metodia ovat *Page*-luokan virtuaaliset *OnNavigatedFrom*- ja *OnNavigatedTo*-metodit, jotka kehittäjä voi ylikirjoittaa omalla toteutuksellaan. Näistä edellistä sovelluskehikko kutsuu aina, kun sovellus siirtyy passiiviseen tilaan, ja tässä metodissa toteutetaankin tilan tallennus. Jälkimmäistä kutsutaan vastaavasti, kun sovellus palaa takaisin aktiiviseen suoritukseen. Tähän metodiin sijoitetaan tilan palautuskoodi.

```
protected override void
    OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    State["SelectedPivotIndex"] = MainPagePivot.SelectedIndex;
}
```

Esimerkki 15. Esimerkkikoodi sovelluksen tilan tallennuksesta PhoneApplicationPage.State -muuttujaan. Koodissa tallennetaan ko. hetkellä valittuna oleva indeksi Pivot-käyttöliittymäkomponentissa.

Tilan tallennuksessa hyödynnetään *PhoneApplicationPage*-luokan (josta kaikki Silverlight for Windows Phone -sovelluksen näkymät periytyvät) *State*-property:a. *State* on tavallinen .NET-tietorakenne, jonka alkiot sovellusalusta tallentaa automaattisesti sovelluksen eristettyyn tietovarastoon, kun sovellus deaktivoituu. *State*-kokoelmaan voi tallentaa mitä tahansa sarjoittuvia (serializable) olioita. Yksittäinen näkymä voi kuitenkin tallentaa maksimissaan 2MB tietoa; koko sovelluksellakin sitä on käytössä ainoastaan 4MB. State-kokoelmaa ei voi myöskään käyttää pitkäaikaisen tiedon, jonka tarkoitus on säilyä sovellusinstanssista toiseen, tallentamiseen vaan tällaisen tiedon tallentamiseen tulee käyttää sovelluksen eristettyä tietovarastoa (Isolated Storage). Eristetty tietovarasto on käsitelty tarkemmin luvussa 6.6.

```
protected override void
    OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    MainPagePivot.SelectedIndex = (int)State["SelectedPivotIndex"];
}
```

Esimerkki 16. Tilan palautus Esimerkki 14 tilan perusteella.

Täydellisen tilansäilytyksen toteuttaminen, jossa sovellus säilyttäisi kaiken tilan vierityspalkkien asemasta tekstilaatikoiden sisällön valittuna olevaan tekstiosuuteen, on tällä hetkellä erittäin haasteellinen ja työläs tehtävä. Jos tilanhallinnasta tinkii hieman

keskittyen hallitsemaan vain käyttäjän kannalta oleellisimpia tilatietoja, voi kehitysajassa säästää huomattavasti aikaa.

6.8 Kokeiluversion toteuttaminen käytännössä

Phone 7 Market Place tarjoaa kehittäjälle mahdollisuuden tukea sovelluksessaan kokeiluversiointia, joka sallii Market Placen asiakkaan ladata ilmaisen trial-version sovelluksesta ennen ostopäätöksen tekemistä. Kehittäjä voi itse suunnitella kokeiluversion sisällön ja esim. jättää osan sovelluksen ominaisuuksista pois kokeiluversiosta. Hyvällä kokeiluversiostrategialla voi parhaimmillaan moninkertaistaa oman sovelluksen myynnin.

Kokeiluversiota varten kehittäjän ei tarvitse luoda uutta versiota sovelluskoodista, vaan toiminnallisuus voidaan integroida suoraan sovelluksen koodiin Trial API:n avulla. Kehittäjä voi Trial API:n avulla myös tarjota suoran ostolinkin sovellukselle, jolloin ostoksen voi tehdä muutamalla painalluksella sovelluksen sisältä.

Kokeilumahdollisuutta tukevaa sovellusta Market Place:n ladattaessa on kuitenkin muistettava merkitä rasti toimituslomakkeen ”Trial Application” -ruutuun.

Ohjelmakoodissa sovelluksen trial-statuksen voi koska tahansa tarkistaa *Microsoft.Phone.MarketPlace* -nimiavaruuden *LicenseInformation*-olion avulla.

```
var license = new LicenseInformation();
bool isTrial = license.IsTrial();
```

Esimerkki 17. Trial-statuksen tarkistaminen LicenseInformation-olion avulla.

Trial-statusta tarkistettaessa on kuitenkin huomioitava verkkoliikenteen viive, koska tarkistaminen luetaan aina verkon yli Market Placen tietokannasta. Hyvä käytäntö onkin lukea heti sovelluksen avautuessa trial-tieto puhelimen muistiin, niin kutsuja Market Placeen ei tarvitse enää tämän jälkeen tehdä.

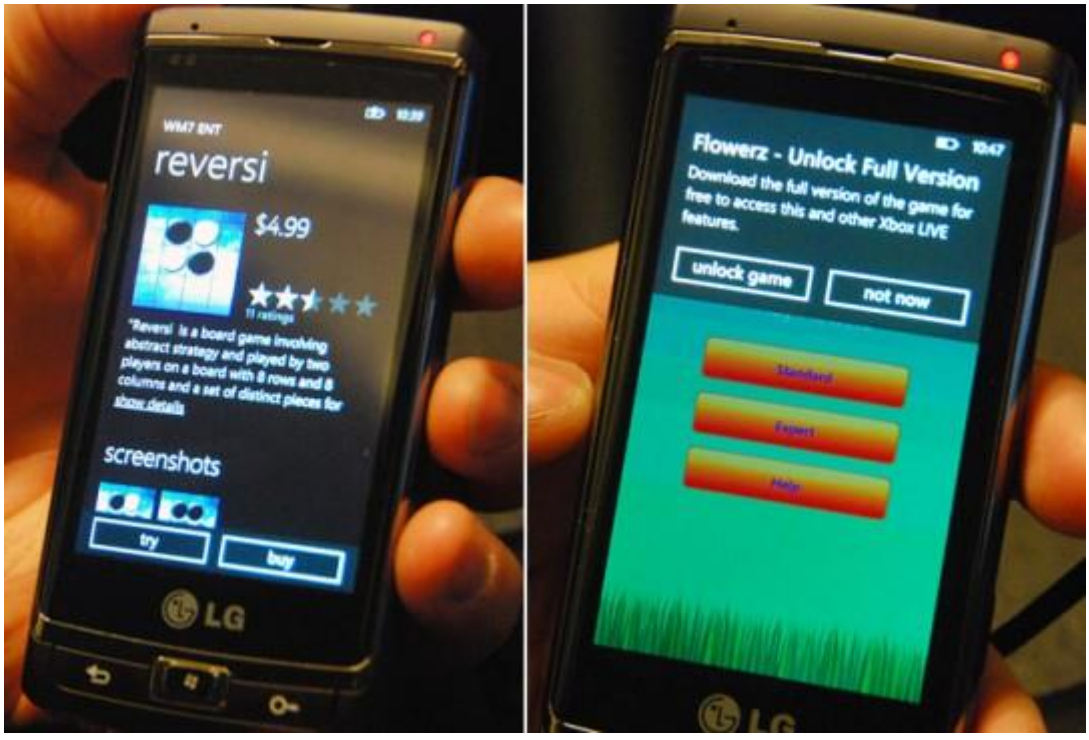
6.8.1 MarketPlaceDetailTask-ostolinkin upottaminen omaan sovellukseen

Trial API mahdollistaa kehittäjän luoda asiakkaalleen nopean ja helpon ostotavan MarketPlaceDetailTask-tehtäväkäynnistäjän avulla. Olion ContentIdentifier-propertyn voi huoletta jättää tyhjäksi, jolloin task näyttää automaattisesti sovelluksen ostosivun, josta tehtävä laukaistiin.

```
MarketplaceDetailTask detailTask = new MarketplaceDetailTask();
detailTask.Show();
```

Esimerkki 18. MarketPlaceDetailTask

*MarketPlaceDetailTask*in *Show*-metodi vie käyttäjän sovelluksen Market Place -sivulle, jonne sovelluksen omistaja voi ladata lisätietoa ja kuvia sovelluksesta. Sivulta löytyy myös painike, jolla ostamisen voi suorittaa.



Kuva 16. Esimerkki sovelluksen ostolinkistä ja Market Place -esittelysivusta.

Trial API ei vielä tällä hetkellä tue skenaarioita, joissa sovelluksen omistaja voisi tarjota asiakkaille kokeiluversion rajatun käyttöajan. On toki helppo tallentaa sovelluksen käyt-

töönottoaika sovelluksen tietovarastoon ja tarkistaa tätä tietoa vasten kokeiluversion käytössäoloaika, mutta tämäkin aiheuttaisi asiakkaalle vain pienen lisäkiusan sovelluksen uudelleenlataamisen muodossa aina, kun aika umpeutuu. Tällaisiin skenaarioihin pitäisikin tuoda mukaan oma palvelin, jonne ladattujen kokeiluversioiden tiedot tallennettaisiin.

7 Yhteenveto

Microsoft on jo pitkään pärjännyt kilpailijoitaan vastaan muissa teknologioissa erityisen helppokäyttöisillä sovelluskehitystyökaluillaan ja ohjelmointikehikoilla, ja Microsoft Windows Phone 7 ei tee tässä asiassa poikkeusta. Yhdessä Microsoft Visual Studio 2010 Express for Windows Phone ja Expression Blend tarjoavat kehittäjille ammattimaisen kehityskokemuksen, joka ei kalpene kilpailijoiden tarjoamille työkaluille. Kehitystyökalut ovat myös kehittäjille ilmaisia, ja koko työkalupaletti tarjoillaan yhdessä internetistä ladattavassa asennuspaketissa.

Kehitystyökalujen laatu ja kivuton asennettavuus onkin yksi Microsoftin valteista sodassa, missä käyttöjärjestelmät pyrkivät saamaan taakseen mahdollisimman suuren kehittäjävoiman.

Kuva 17. Market Placen tarjolla olevien sovellusten määrä. 10 000 sovelluksen raja rikkoontui 11.3.2011. [17.]

Microsoft Phone 7:n tiukka laitestandardointi ennaltaehkäisee laitekannan pirstaloitumisongelmaa, jossa kehittäjän on hyvin vaikea tehdä sovellusta joka toimisi kaikissa mahdollisissa käyttöjärjestelmää hyödyntävistä laitteista. Toisaalta Phone 7 -laitteita voi silti valmistaa useampi kuin yksi valmistaja, jolloin laitteiden potentiaalinen valmistusnopeus ja saatavilla olevien mallien määrä on korkeampi kuin yhden valmistajan alustoissa.

Windows Phone 7 -päätelaitteen tarjoamia palveluita ja sensoreita on kehittäjän helppo hyödyntää yksinkertaisten ja yhteneväisten ohjelmointirajapintojen avulla. Toisaalta mm. socket-tuen puute (tulossa myöhemmin) ja aito moniajo kolmannen osapuolen sovelluksissa asettavat tällä hetkellä rajoja tietyntyyppisten sovellusten kehittämiseen. Myös tombstoning, sovelluksen tilan tallennus ennen sovelluksen deaktivoimista ja palautus deaktivoinnin jälkeen, aiheuttaa kehittäjälle tällä hetkellä paljon manuaalista työtä. Kehittäjän on myös osittain haasteellista tehdä käyttöjärjestelmään istuvaa käyttöliittymää valmiiden käyttöliittymäkomponenttien nykyisellä tarjonnalla.

Kehittäjän on tällä hetkellä mahdollista valita kahdesta täysin erilaisesta sovellusalustasta, Silverlight for Windows Phone ja XNA Game Studio 4.0, jotka kumpikin täyttävät oman erityisen tehtävänsä mobiilisovellusalustoina ja kummankin olemassaolo on perusteltua Silverlightin tarjotessa alustan mediasovelluksille, ja XNA:n mobiilipelikehitykseen.

Market Place on sovelluskehittäjälle helppo tapa levittää sovelluksensa ympäri maailman kaikkien Phone 7 -laitteiden omistajien ulottuville. Kauppapaikan sertifiointiprosessi on riittävän läpinäkyvä tarjoten kehittäjälle avointa dokumentaatiota sovelluksen vaatimuksista. Kauppapaikka myös antaa kehittäjälle takaisin laadukkaan raportin testausprosessin aikana mahdollisesti löytyneistä puutteista, jolloin kehittäjän on helppo paikata puutteet ja saada sovelluksensa sertifioitua. Kehittäjälle tarjotaan myös valmista ohjelmointirajapintaa kokeiluversioiden tekoon, ja työkalujen avulla on myös mahdollista tarjota suoraa linkkiä sovelluksen ostosivulle.

Lähteet

1 Windows CE. Verkkodokumentti. <http://en.wikipedia.org/wiki/Windows_CE>.

2 Windows Mobile. Verkkodokumentti. <http://en.wikipedia.org/wiki/Windows_Mobile>.

3 Windows Phone 7. Verkkodokumentti. <http://en.wikipedia.org/wiki/Windows_Phone_7>.

4 Highlights of XNA development on Windows Phone 7. Verkkodokumentti. <http://mobile.dzone.com/articles/xna-development-windows-phone>.

5 How to: Preserve and Restore Page State for Windows Phone. Verkkodokumentti. <http://msdn.microsoft.com/en-us/library/ff967548(v=vs.92).aspx>.

6 Introducing Windows Phone 7. Verkkodokumentti. <http://msdn.microsoft.com/en-us/library/gg490768.aspx>.

7 Features Supported in Silverlight for Windows Phone. Verkkodokumentti. <http://msdn.microsoft.com/en-us/library/ff426931%28VS.96%29.aspx>.

8 Pocket PC. Verkkodokumentti. <http://en.wikipedia.org/wiki/Pocket_PC>.

9 The Windows Phone Developer Blog. Verkkodokumentti. <http://windowsteamblog.com/windows_phone/b/wpdev/archive/2010/07/15/understanding-the-windows-phone-application-execution-model-tombstoning-launcher-and-choosers-and-few-more-things-that-are-on-the-way-part-1.aspx>.

10 Phone 7 Simulates Continuity. Verkkodokumentti. <http://www.informationweek.com/news/windows/operatingsystems/showArticle.jhtml?articleID=229200269>.

11 List of Windows Phone devices. Verkkodokumentti. <http://en.wikipedia.org/wiki/List_of_Windows_Phone_devices>.

12 UI Design and Interaction Guide for Windows Phone 7. 2010. Verkkodokumentti. Microsoft.

13 Reittiopas API. Verkkodokumentti. Helsingin seudun liikenne. <http://developer.reittiopas.fi/pages/fi/reittiopas-api.php>.

14 Designing Applications for Windows Phone. Verkkodokumentti. <http://msdn.microsoft.com/en-us/library/gg490770.aspx>.

15 App Hub. Verkkodokumentti. <http://create.msdn.com>.

16 Windows Phone 7: the complete guide. Verkkodokumentti. <http://www.engadget.com/2010/03/18/windows-phone-7-series-the-complete-guide>.

17 Windows Phone Applist. Verkkodokumentti. <http://www.windowsphoneapplist.com>.

Silverlight EllipseButtonStyle

```
<Style x:Key="EllipseButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="BorderBrush" Value="{StaticResource PhoneForegroundBrush}"/>
            <Setter Property="Foreground" Value="{StaticResource PhoneForegroundBrush}"/>
            <Setter Property="BorderThickness" Value="{StaticResource PhoneBorderThickness}"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Grid Background="Transparent">
                            <VisualStateManager.VisualStateGroups>
                                <VisualStateGroup x:Name="CommonStates">
                                    <VisualState x:Name="Normal"/>
                                    <VisualState x:Name="MouseOver"/>
                                    <VisualState x:Name="Pressed">
                                        <Storyboard>
                                            <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Fill"
                                                Storyboard.TargetName="ButtonBackground">
                                                <DiscreteObjectKeyFrame KeyTime="0"
                                                     Value="{StaticResource PhoneBackgroundBrush}"/>
                                            </ObjectAnimationUsingKeyFrames>
                                            <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Stroke"
                                                 Storyboard.TargetName="ButtonBackground">
                                                <DiscreteObjectKeyFrame KeyTime="0"
                                                    Value="{StaticResource PhoneBackgroundBrush}"/>
                                            </ObjectAnimationUsingKeyFrames>
                                            <ObjectAnimationUsingKeyFrames
                                                Storyboard.TargetProperty="Fill"Storyboard.TargetName="ButtonBackground2">
                                                <DiscreteObjectKeyFrame KeyTime="0"
                                                    Value="{StaticResource PhoneForegroundBrush}"/>
                                            </ObjectAnimationUsingKeyFrames>
                                            <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Stroke"
                                                Storyboard.TargetName="ButtonBackground2">
                                                <DiscreteObjectKeyFrame KeyTime="0"
                                                    Value="{StaticResource PhoneForegroundBrush}"/>
                                            </ObjectAnimationUsingKeyFrames>
                                        </Storyboard>
                                    </VisualState>
                                    <VisualState x:Name="Disabled">
                                        <Storyboard>
                                            <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Stroke"
                                                Storyboard.TargetName="ButtonBackground">
                                                <DiscreteObjectKeyFrame KeyTime="0"
                                                    Value="{StaticResource PhoneDisabledBrush}"/>
                                            </ObjectAnimationUsingKeyFrames>
                                            <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Background"
                                                Storyboard.TargetName="ButtonBackground">
                                                <DiscreteObjectKeyFrame KeyTime="0" Value="Transparent"/>
                                            </ObjectAnimationUsingKeyFrames>
                                        </Storyboard>
                                    </VisualState>
                                </VisualStateGroup>
                            </VisualStateManager.VisualStateGroups>
                            <Ellipse x:Name="ButtonBackground2" HorizontalAlignment="Center" Height="45" Width="45"
                                 Margin="{StaticResource PhoneTouchTargetOverhang}"
                                 Stroke="{TemplateBinding BorderBrush}" VerticalAlignment="Center"
                                 StrokeThickness="3" Fill="{StaticResource PhoneSemitransparentBrush}">
                            </Ellipse>
                            <Ellipse x:Name="ButtonBackground" OpacityMask="{TemplateBinding Content}"
                                 HorizontalAlignment="Center" Height="45" Width="45"
                                 Margin="{StaticResource PhoneTouchTargetOverhang}" Stroke="{TemplateBinding BorderBrush}"
                                 VerticalAlignment="Center" StrokeThickness="3"
                                 Fill="{StaticResource PhoneForegroundBrush}">
                            </Ellipse>
                        </Grid>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
```

Lyhenteitä ja käsitteitä

.NET tai .NET Framework	Microsoftin kehittämä abstraktiokerros ja virtuaalinen ajoympäristö Windows -käyttöjärjestelmän päälle.
Android	Google:n käyttöjärjestelmä mobiililaitteille.
C#	Microsoftin luoma ohjelmointikieli .NET -sovelluskehikolle.
CSS	Cascading Style Sheets. Käytetään kuvaamaan dokumentin esityssemantiikkaa, erityisesti web-maailmassa.
DirectX	Microsoftin määrittelemä, erityisesti peliohjelmointiin suunnattu rajapinta sovelluksen ja laitteiston välille.
GPS	Global Positioning System.
HLSL	High Level Shader Language. Direct 3D:n kanssa toimiva pixel shader -ohjelmointikieli.
iOS	Apple:n käyttöjärjestelmä mobiililaitteille.
Kapasitiivinen kosketusnäyttö	Tekniikka, jossa kosketuksen tunnistus perustuu kosketuspään tuottamaan muutokseen näytön sähköstaattisessa kentässä.
Silverlight	Microsoftin sovelluskehikko rikkaille mediasovelluksille.
TFS	Team Foundation Server. Microsoftin versionhallintajärjestelmä.
XAML	Extensible Application Markup Language. Microsoftin määrittelemä kuvauskieli erityisesti käyttöliittymien kuvaamiseen.
XNA	Microsoftin videopelikehitykseen suunnattu sovelluskehikko.